Welcome to Covid19 Data Analysis Notebook

Let's Import the modules

In [50]:
```
%pip install seaborn
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
print('Modules are imported.')
```

```
Modules are imported.
```

Task 2

Task 2.1: importing covid19 dataset

importing "Covid19_Confirmed_dataset.csv" from "./Dataset" folder.

In [2]:
```
Corona_Data = pd.read_csv(r'covid19_Confirmed_dataset.csv')
Corona_Data
```

Out[2]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20
0	NaN	Afghanistan	33.000000	65.000000	0	0	0	0	0
1	NaN	Albania	41.153300	20.168300	0	0	0	0	0
2	NaN	Algeria	28.033900	1.659600	0	0	0	0	0
3	NaN	Andorra	42.506300	1.521800	0	0	0	0	0
4	NaN	Angola	-11.202700	17.873900	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
261	NaN	Western Sahara	24.215500	-12.885800	0	0	0	0	0
262	NaN	Sao Tome and Principe	0.186360	6.613081	0	0	0	0	0
263	NaN	Yemen	15.552727	48.516388	0	0	0	0	0
264	NaN	Comoros	-11.645500	43.333300	0	0	0	0	0
265	NaN	Tajikistan	38.861034	71.276093	0	0	0	0	0

266 rows × 104 columns

Let's check the shape of the dataframe

In [3]:
```
Corona_Data.shape
```

Out[3]: (266, 104)

Task 2.2: Delete the useless columns

In [4]:
```
Corona_Data.drop(['Lat', 'Long'], axis = 1, inplace = True)
Corona_Data.head()
```

Out[4]:

	Province/State	Country/Region	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29
0	NaN	Afghanistan	0	0	0	0	0	0	0	
1	NaN	Albania	0	0	0	0	0	0	0	
2	NaN	Algeria	0	0	0	0	0	0	0	
3	NaN	Andorra	0	0	0	0	0	0	0	
4	NaN	Angola	0	0	0	0	0	0	0	

5 rows × 102 columns

In [ ]:
```
```

Task 2.3: Aggregating the rows by the country

In [5]:
```
Corona_Data_Agg = Corona_Data.groupby('Country/Region').sum()
```

```
<ipython-input-5-f51f54de0d4e>:1: FutureWarning: The default value of numeric_only in Data
FrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. E
ither specify numeric_only or select only columns which should be valid for the function.
  Corona_Data_Agg = Corona_Data.groupby('Country/Region').sum()
```

In [6]:
```
Corona_Data_Agg.head()
```

Out[6]:

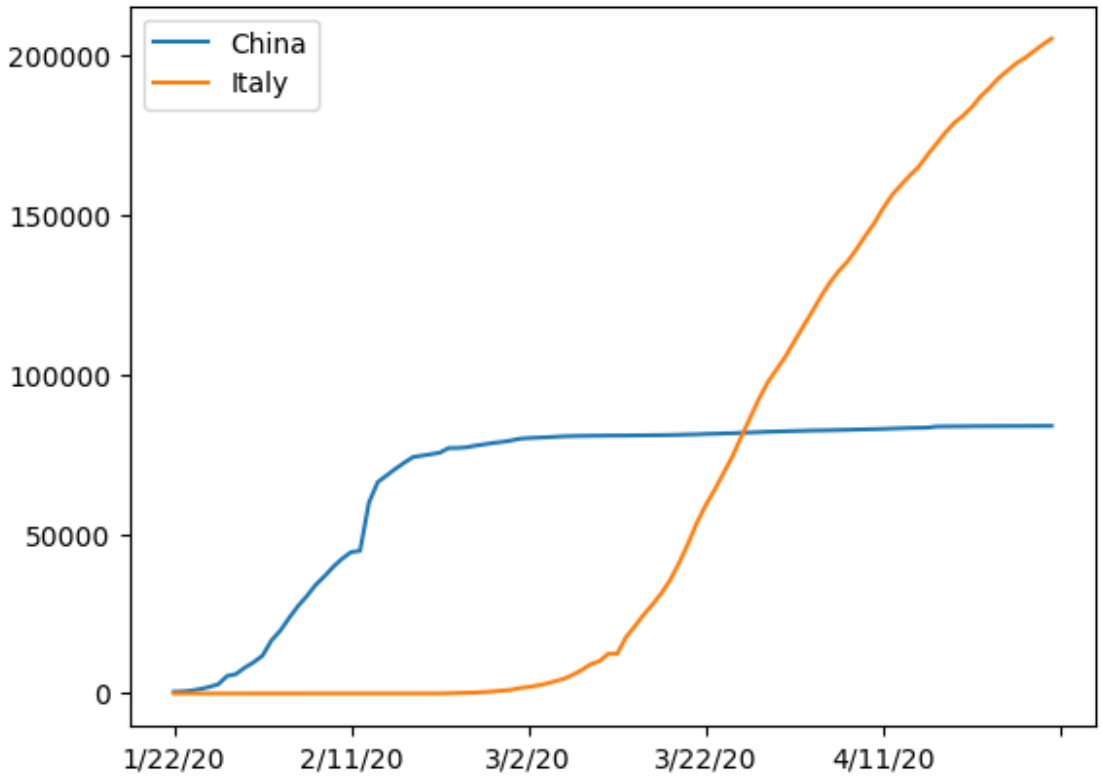
Country/Region	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31
Afghanistan	0	0	0	0	0	0	0	0	0	
Albania	0	0	0	0	0	0	0	0	0	
Algeria	0	0	0	0	0	0	0	0	0	
Andorra	0	0	0	0	0	0	0	0	0	
Angola	0	0	0	0	0	0	0	0	0	

5 rows × 100 columns

In [16]:
```
Corona_Data_Agg.loc['China'].plot()
Corona_Data_Agg.loc['Italy'].plot()
```

```
plt.legend()
```

Out[16]: <matplotlib.legend.Legend at 0x5489368>



Task 2.4: Visualizing data related to a country for example China

visualization always helps for better understanding of our data.

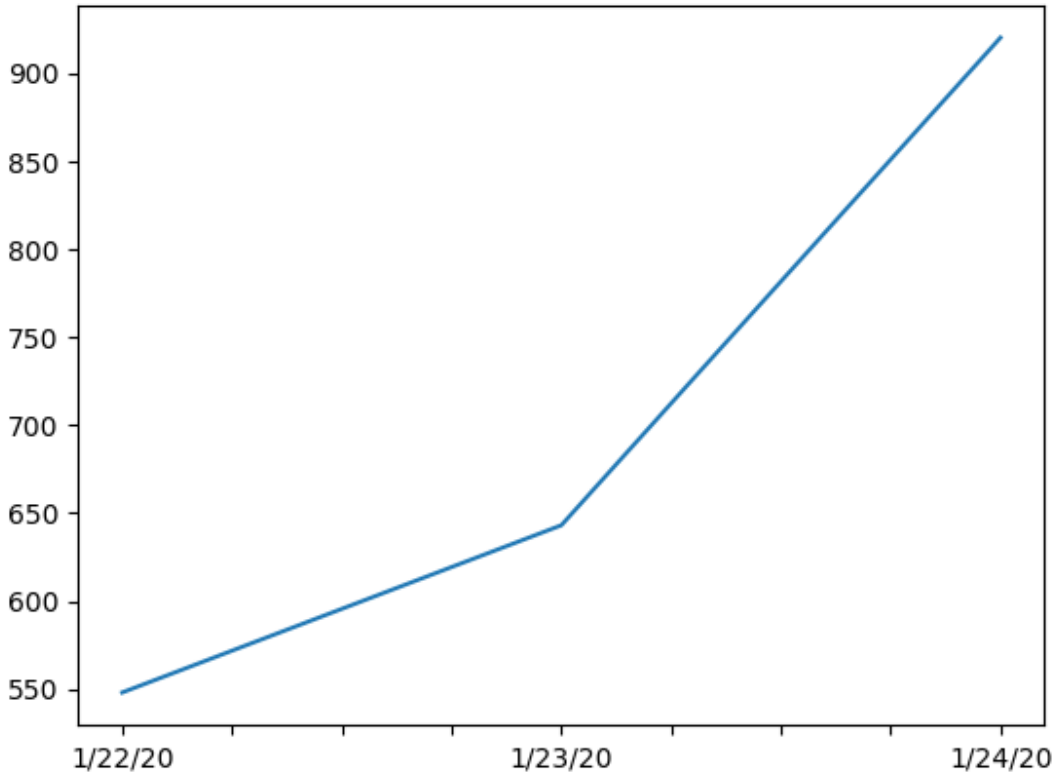
In [ ]:
```
Corona_Data_Agg.loc['China'].plot()
```

Task3: Calculating a good measure

we need to find a good measure reperestend as a number, describing the spread of the virus in a country.

In [8]:
```
Corona_Data_Agg.loc['China'][:3].plot()
```

Out[8]: <AxesSubplot:>

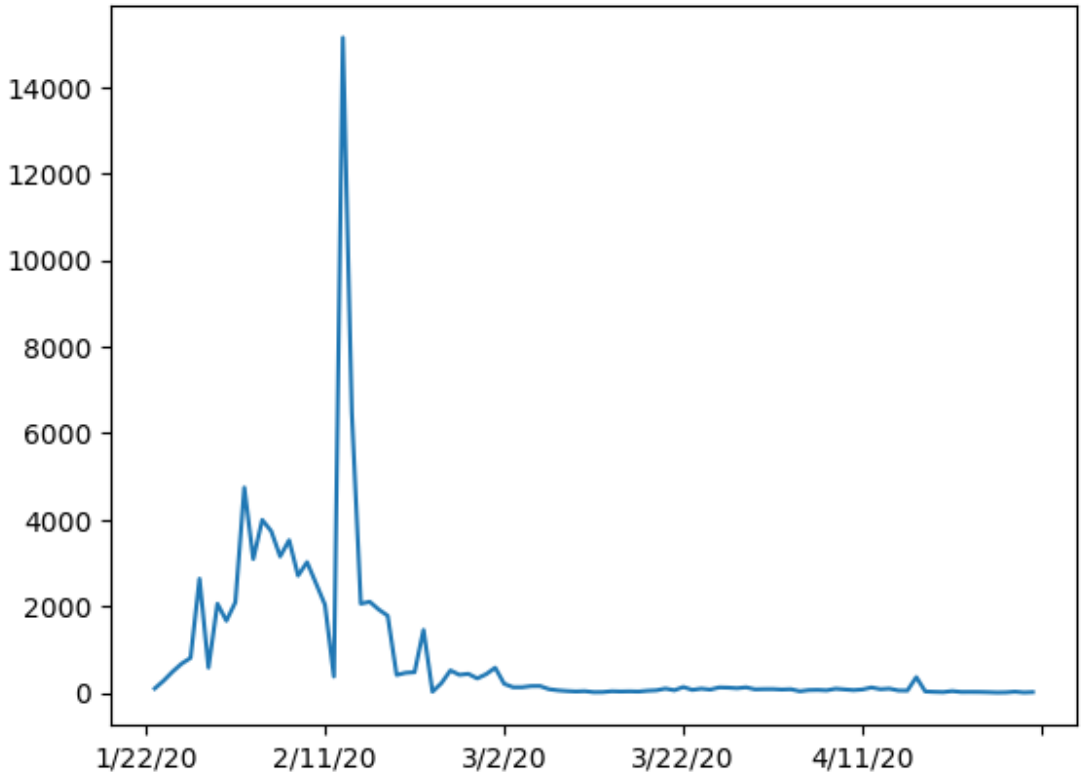


In [ ]:

task 3.1: caculating the first derivative of the curve

In [9]:
```
Corona_Data_Agg.loc['China'].diff().plot()
```

Out[9]: <AxesSubplot:>



task 3.2: find maxmimum infection rate for China

In [10]:
```
Corona_Data_Agg.loc['China'].diff().max()
```

Out[10]:
```
15136.0
```

In [11]:
```

```

In [ ]:
```
Countries_max_infected = []
for c in Countries:
    Countries_max_infected.append(Corona_Data_Agg.loc[c].diff().max())
```

Task 3.3: find maximum infection rate for all of the countries.

In [12]:
```
Countries = list(Corona_Data_Agg.index)
```

In [14]:
```
Countries_max_infected = []
for c in Countries:
    Countries_max_infected.append(Corona_Data_Agg.loc[c].diff().max())
Corona_Data_Agg['Countries_max_infected'] = Countries_max_infected
Corona_Data_Agg.head()
```

Out[14]:

Country/Region	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31
Afghanistan	0	0	0	0	0	0	0	0	0	
Albania	0	0	0	0	0	0	0	0	0	
Algeria	0	0	0	0	0	0	0	0	0	
Andorra	0	0	0	0	0	0	0	0	0	
Angola	0	0	0	0	0	0	0	0	0	

5 rows × 101 columns

Task 3.4: create a new dataframe with only needed column

In [21]:
```
New_Data = pd.DataFrame(Corona_Data_Agg['Countries_max_infected'])
New_Data.head()
```

Out[21]:

Country/Region	Countries_max_infected
Afghanistan	232.0
Albania	34.0
Algeria	199.0
Andorra	43.0
Angola	5.0

In [22]:
```
New_Data.shape
```

Out[22]:
```
(187, 1)
```

Task4:

- Importing the WorldHappinessReport.csv dataset
- selecting needed columns for our analysis
- join the datasets
- calculate the correlations as the result of our analysis

Task 4.1 : importing the dataset

In [23]:
```
WH_Data = pd.read_csv(r'worldwide_happiness_report.csv')
WH_Data
```

Out[23]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298
...	...	...	...	...	...	...	...	...	...
151	152	Rwanda	3.334	0.359	0.711	0.614	0.555	0.217	0.411
152	153	Tanzania	3.231	0.476	0.885	0.499	0.417	0.276	0.147
153	154	Afghanistan	3.203	0.350	0.517	0.361	0.000	0.158	0.025
154	155	Central African Republic	3.083	0.026	0.000	0.105	0.225	0.235	0.035
155	156	South Sudan	2.853	0.306	0.575	0.295	0.010	0.202	0.091

156 rows × 9 columns

In [ ]:

Task 4.2: let's drop the useless columns

In [25]:
```
WH_Data.drop(['Overall rank', 'Score','Generosity','Perceptions of corruption'], axis = 1,
```

In [26]:
```
WH_Data
```

Out[26]:

	Country or region	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
0	Finland	1.340	1.587	0.986	0.596
1	Denmark	1.383	1.573	0.996	0.592
2	Norway	1.488	1.582	1.028	0.603
3	Iceland	1.380	1.624	1.026	0.591
4	Netherlands	1.396	1.522	0.999	0.557
...	...	...	...	...	...
151	Rwanda	0.359	0.711	0.614	0.555
152	Tanzania	0.476	0.885	0.499	0.417
153	Afghanistan	0.350	0.517	0.361	0.000
154	Central African Republic	0.026	0.000	0.105	0.225
155	South Sudan	0.306	0.575	0.295	0.010

156 rows × 5 columns

Task 4.3: changing the indices of the dataframe

In [35]:
```
WH_Data.set_index(['Country or region'], inplace = True)
WH_Data.head()
```

Out[35]:

Country or region	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
Finland	1.340	1.587	0.986	0.596
Denmark	1.383	1.573	0.996	0.592
Norway	1.488	1.582	1.028	0.603
Iceland	1.380	1.624	1.026	0.591
Netherlands	1.396	1.522	0.999	0.557

Task4.4: now let's join two dataset we have prepared

Corona Dataset :

wolrd happiness report Dataset :

In [36]:
```
New_Data.head()
```

Out[36]:

Country/Region	Countries_max_infected
Afghanistan	232.0
Albania	34.0
Algeria	199.0
Andorra	43.0
Angola	5.0

In [44]:
```
WH_Data.shape
```

Out[44]:
```
(156, 4)
```

In [42]:
```
New_Data.shape
```

Out[42]:
```
(187, 1)
```

In [46]:
```
CoronaData = New_Data.join(WH_Data, how = 'inner')
CoronaData.head()
```

Out[46]:

	Countries_max_infected	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
Afghanistan	232.0	0.350	0.517	0.361	0.000
Albania	34.0	0.947	0.848	0.874	0.383
Algeria	199.0	1.002	1.160	0.785	0.086
Argentina	291.0	1.092	1.432	0.881	0.471
Armenia	134.0	0.850	1.055	0.815	0.283

Task 4.5: correlation matrix

In [47]:
```
CoronaData.corr()
```

Out[47]:

	Countries_max_infected	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
Countries_max_infected	1.000000	0.250118	0.191958	0.289263	0.078196
GDP per capita	0.250118	1.000000	0.759468	0.863062	0.394603
Social support	0.191958	0.759468	1.000000	0.765286	0.456246
Healthy life expectancy	0.289263	0.863062	0.765286	1.000000	0.427892
Freedom to make life choices	0.078196	0.394603	0.456246	0.427892	1.000000

Task 5: Visualization of the results

our Analysis is not finished unless we visualize the results in terms figures and graphs so that everyone can understand what you get out of our analysis

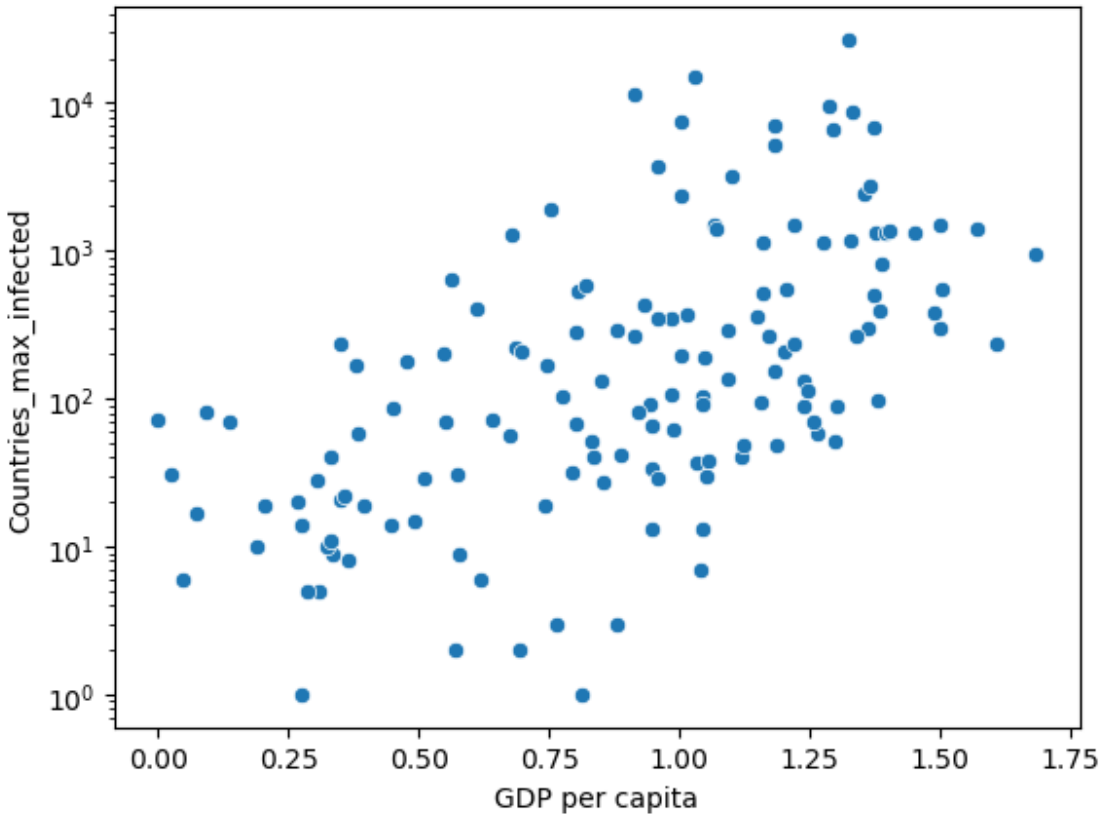
In [48]:
```
CoronaData.head()
```

Out[48]:

	Countries_max_infected	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
Afghanistan	232.0	0.350	0.517	0.361	0.000
Albania	34.0	0.947	0.848	0.874	0.383
Algeria	199.0	1.002	1.160	0.785	0.086
Argentina	291.0	1.092	1.432	0.881	0.471
Armenia	134.0	0.850	1.055	0.815	0.283

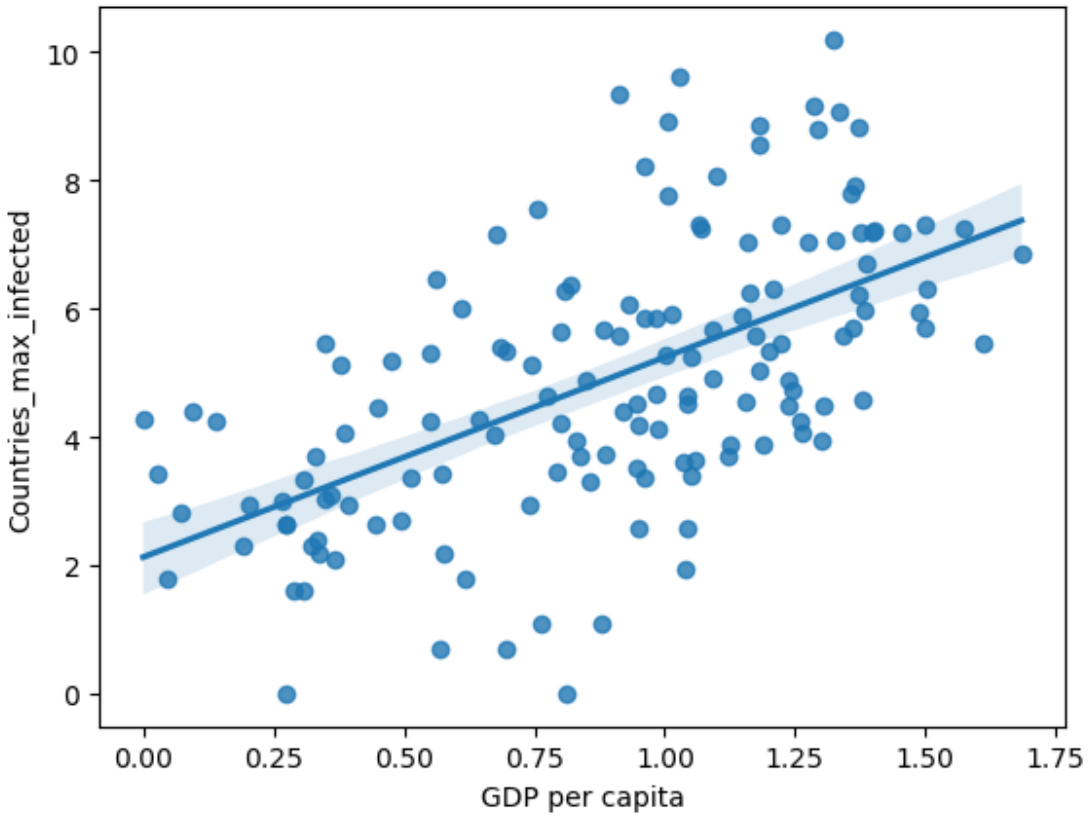
Task 5.1: Plotting GDP vs maximum Infection rate

In [63]:
```
sns.scatterplot(x= "GDP per capita", y = "Countries_max_infected", data = CoronaData)
plt.yscale('log')
```



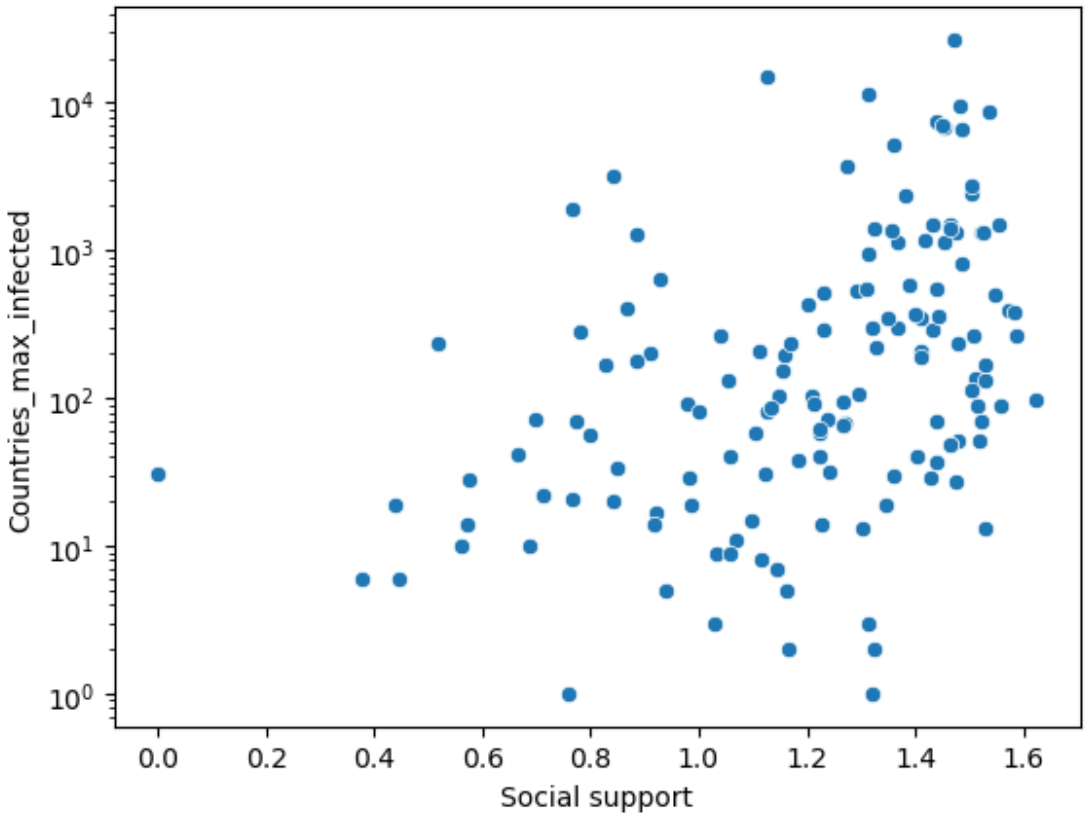
In [69]:
```
sns.regplot(x= CoronaData["GDP per capita"], y = np.log(CoronaData["Countries_max_infected
```

Out[69]:
```
<AxesSubplot:xlabel='GDP per capita', ylabel='Countries_max_infected'>
```



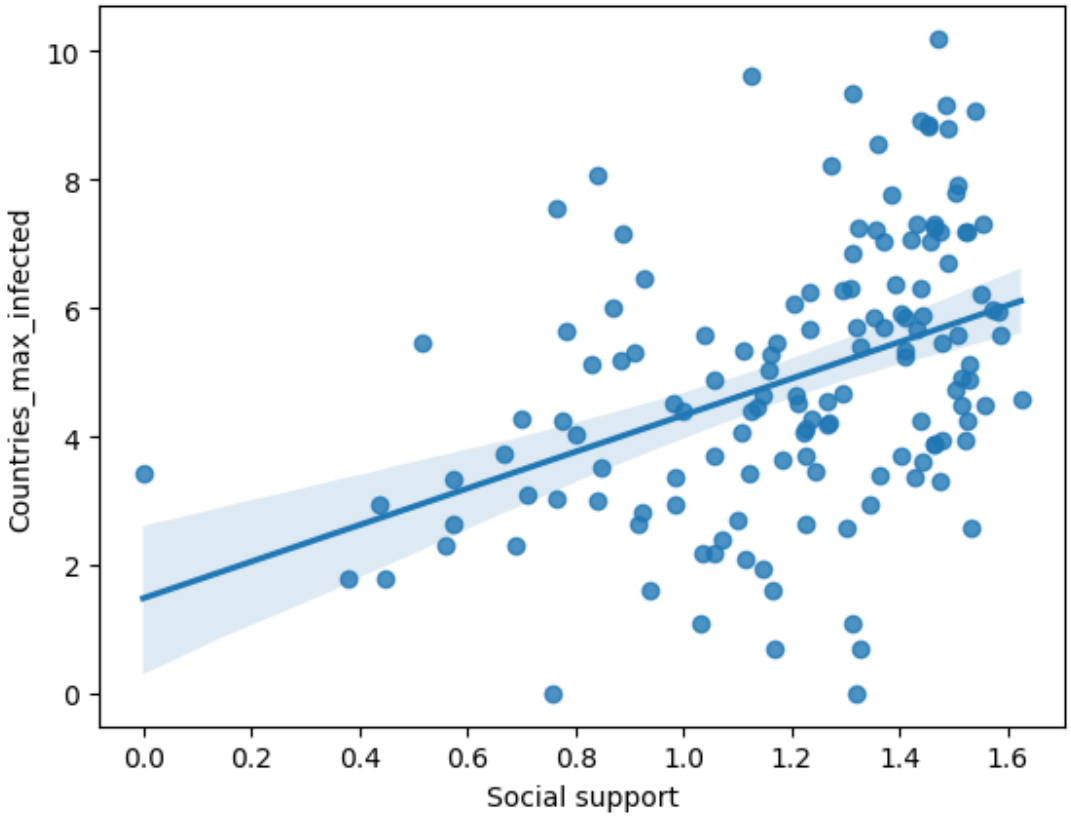
Task 5.2: Plotting Social support vs maximum Infection rate

In [70]:
```
sns.scatterplot(x= "Social support", y = "Countries_max_infected", data = CoronaData)
plt.yscale('log')
```



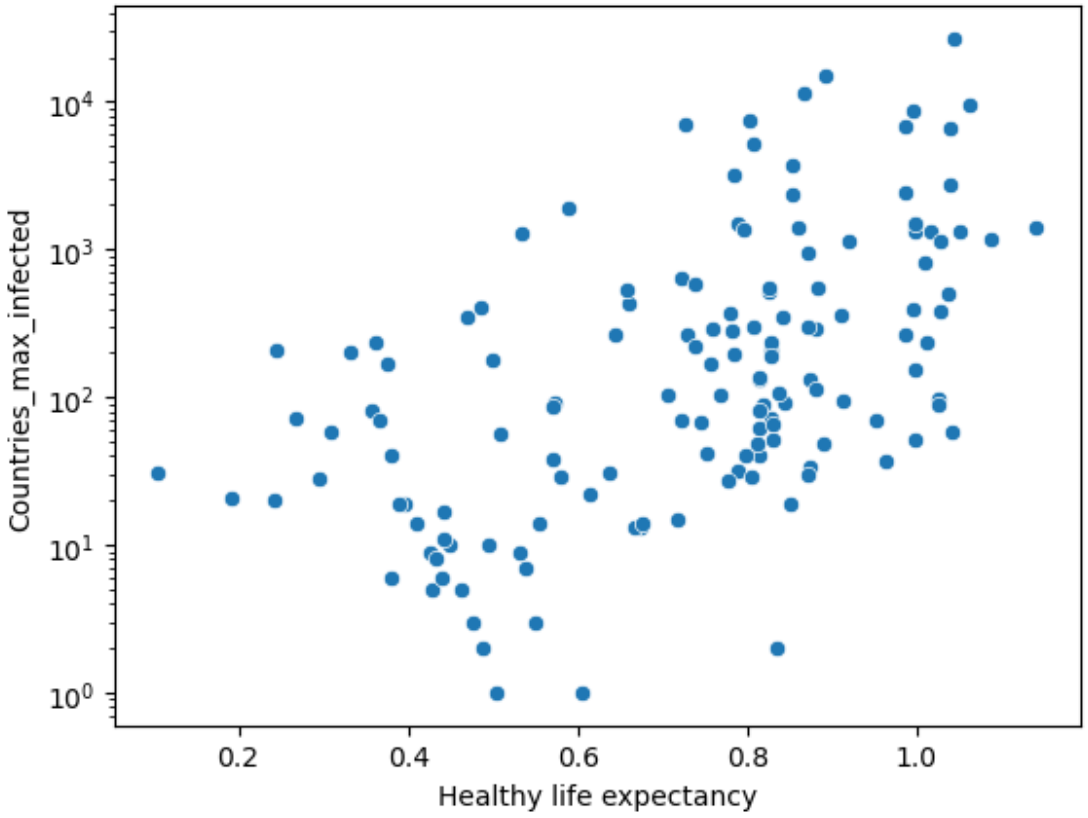
In [71]:
```
sns.regplot(x= CoronaData["Social support"], y = np.log(CoronaData["Countries_max_infected
```

Out[71]:
```
<AxesSubplot:xlabel='Social support', ylabel='Countries_max_infected'>
```



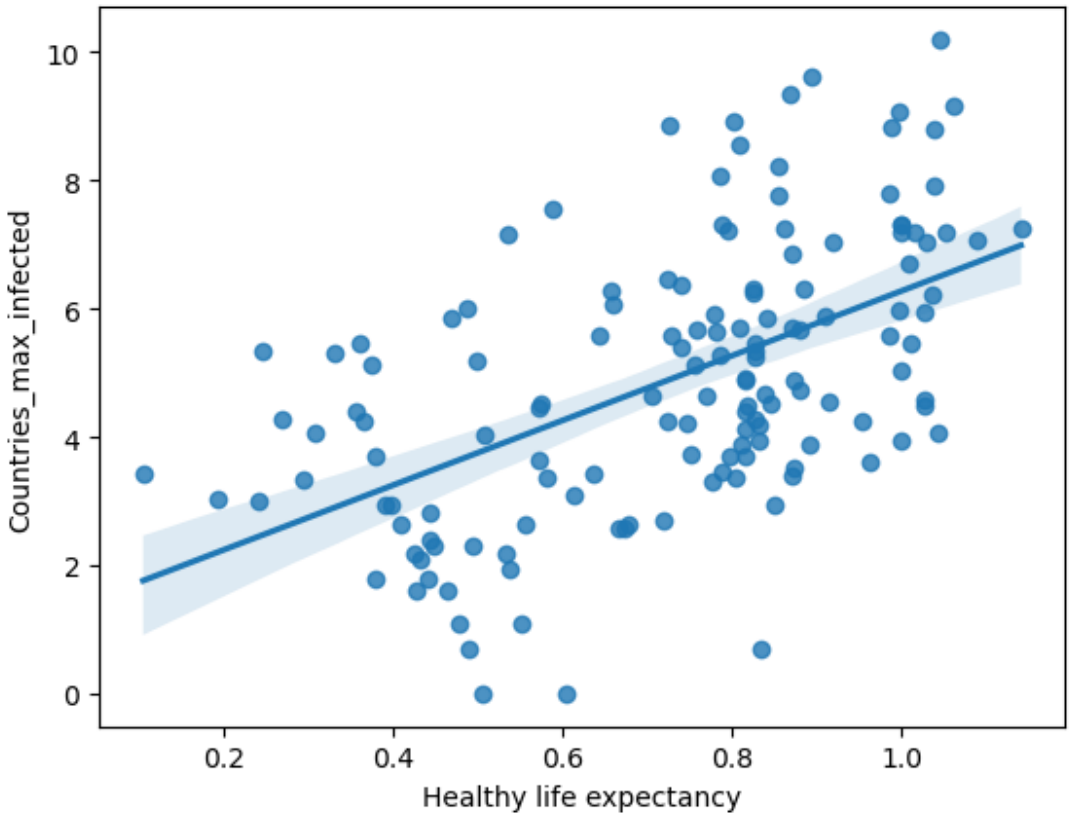
Task 5.3: Plotting Healthy life expectancy vs maximum Infection rate

In [72]:
```
sns.scatterplot(x= "Healthy life expectancy", y = "Countries_max_infected", data = CoronaD
plt.yscale('log')
```



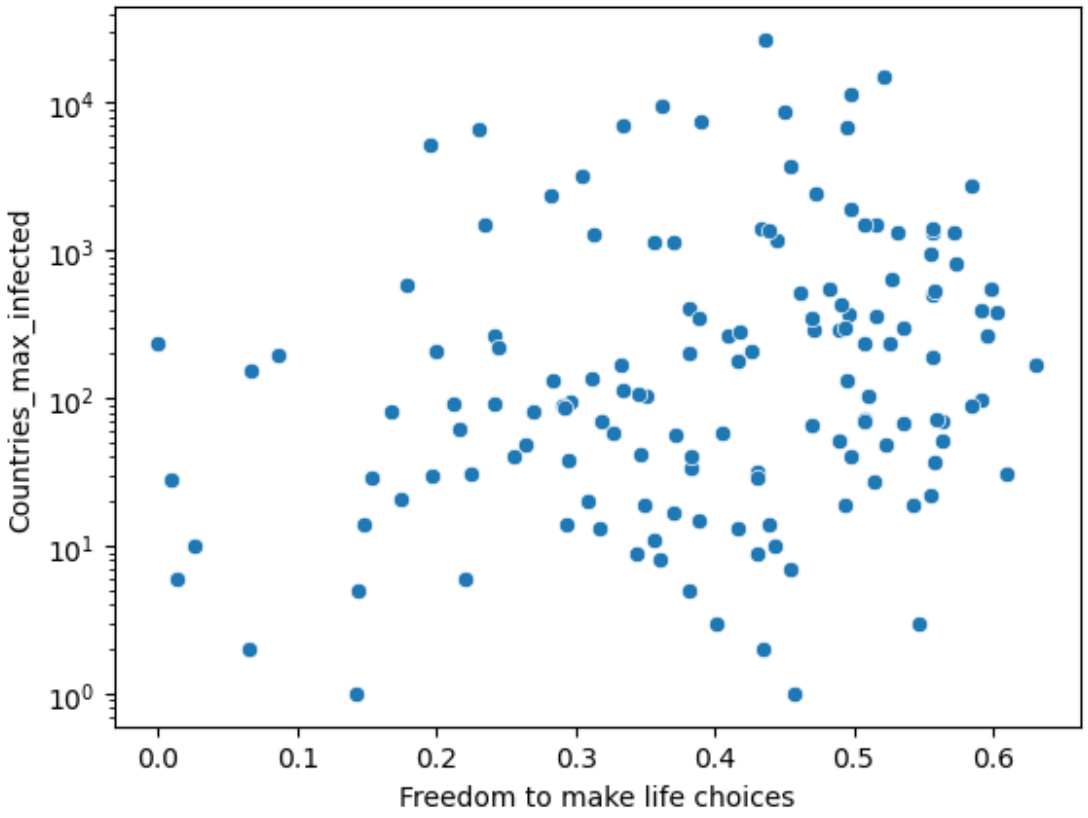
In [73]:
```
sns.regplot(x= CoronaData["Healthy life expectancy"], y = np.log(CoronaData["Countries_max
```

Out[73]:
```
<AxesSubplot:xlabel='Healthy life expectancy', ylabel='Countries_max_infected'>
```



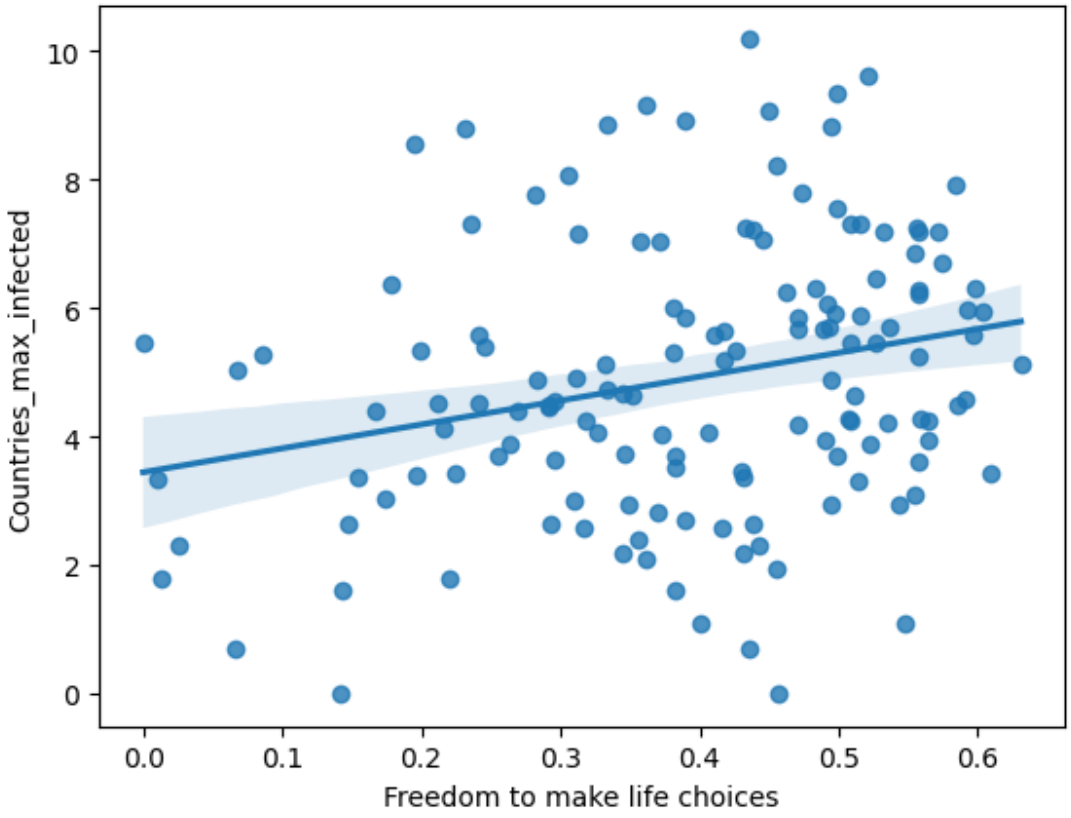
Task 5.4: Plotting Freedom to make life choices vs maximum Infection rate

In [74]:
```
sns.scatterplot(x= "Freedom to make life choices", y = "Countries_max_infected", data = Co
plt.yscale('log')
```



In [75]:
```
sns.regplot(x= CoronaData["Freedom to make life choices"], y = np.log(CoronaData["Countrie
```

Out[75]:
```
<AxesSubplot:xlabel='Freedom to make life choices', ylabel='Countries_max_infected'>
```



In [ ]: